Weekly Work Report

Wenjie Niu

VISION@OUC

August 5, 2018

1 Research problem

Understand the paper and run the code of *Image-to-Image Translation Using Conditional Adversarial Networks(pix2pix)* [1]. Understand the paper of *Discriminative Region Proposal Adversarial Networks for High-Quality Image-to-Image Translation(DRPAN)* [5] correcly.

2 Research approach

Firstly, read the paper of pix2pix, then run the code of it to have an intimate knowledge of the origin model, which be explored in DRPAN. Later, read the paper of DRPAN again for running the its code and expansion of the experiments.

3 Research progress

Finished the Pytorch all introductory tutorials, which includes the basic instruments and programming. Learned the method of constructing a neural network and related knowledge such as loss function, forward and back propagation, updating weight and how to train a classfier. Then use the network of Auto-Encoder to train MNIST with FC layers and CNN layers seperately. Finally, train it with GAN and DCGAN.

4 Progress in this week

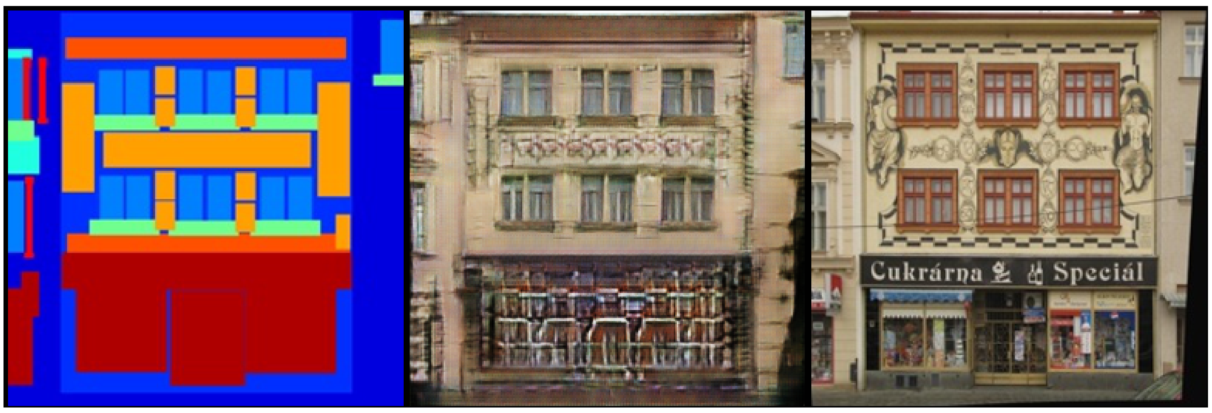

Figure 1: The output of training in the dataset of facades.

Step 1 Read and try to understand the paper of Image-to-Image Translation Using Conditional Adversarial Networks(pix2pix) and some blogs to help with mastering its structure. The architecture of generator is using a U-net as shown in figure 2. Two choices for the architecture of the generator. The U-Net is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks. While the discriminator use a convolutional “PatchGAN” classifier. They adapt our generator and discriminator architectures from those in [4]. Both generator and discriminator use modules of the form convolution-BatchNorm-ReLu [3].

Step 2 Run the code of pix2pix with the dataset of facades, the train output is shown in figure. 1 and the test output is as shown in figure 3 4. The intermediate results are as shown in figure 5 6.

Step 3 At the same time, I try to configuration environment and use GPU to run codes and imitate the basic command on GPU which is really hard to handle for myself.

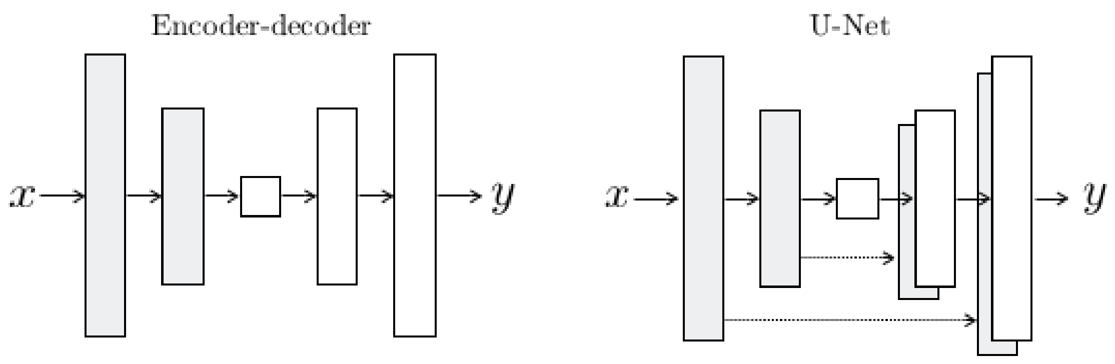

Figure 2: The architecture of generator

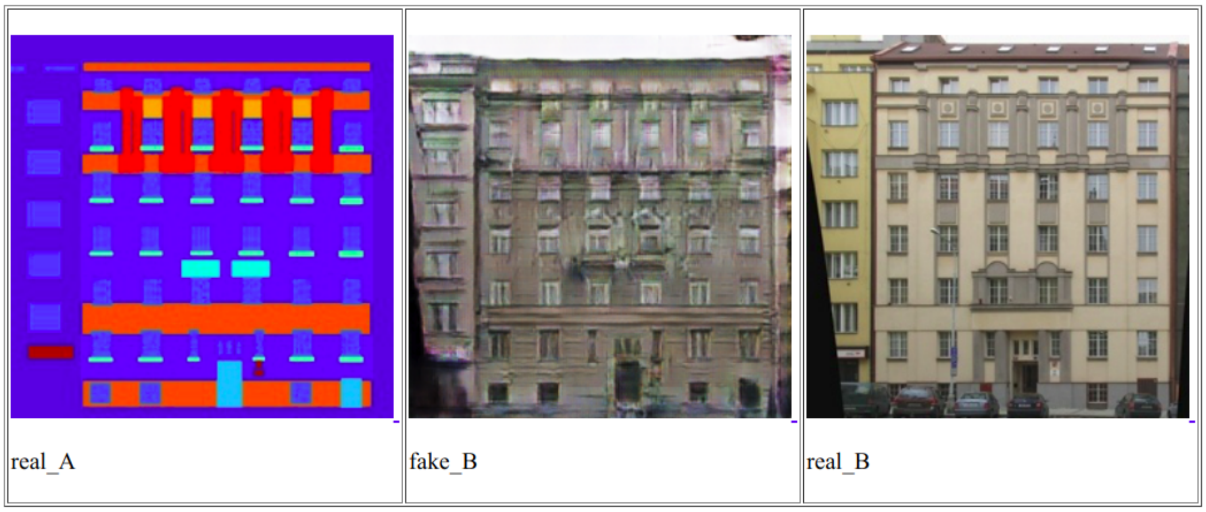

Figure 3: The output of test dataset.

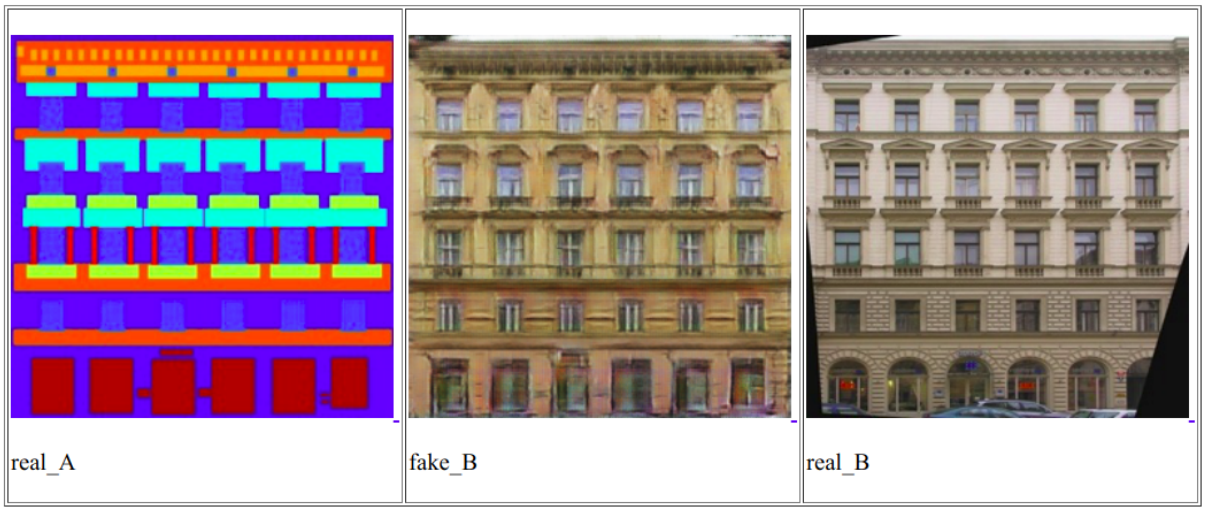

Figure 4: The output of test dataset.

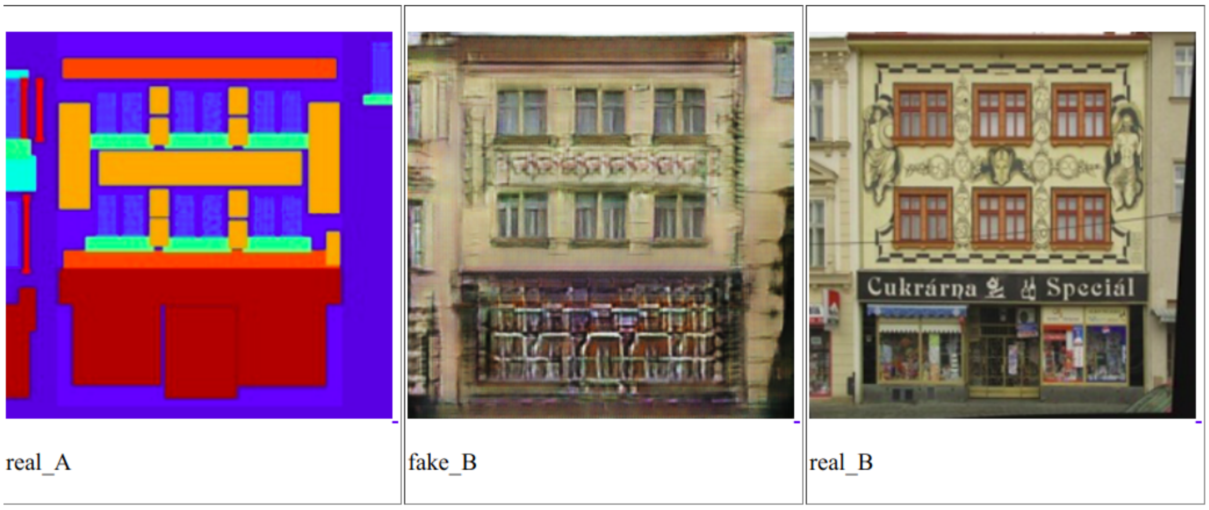

Figure 5: The intermediate results.

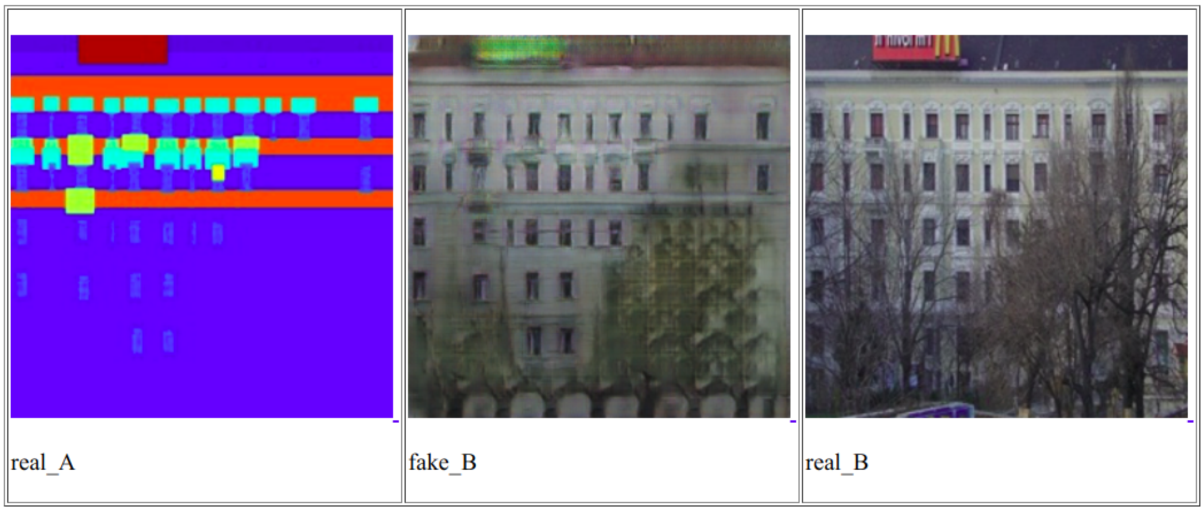

Figure 6: The intermediate results.

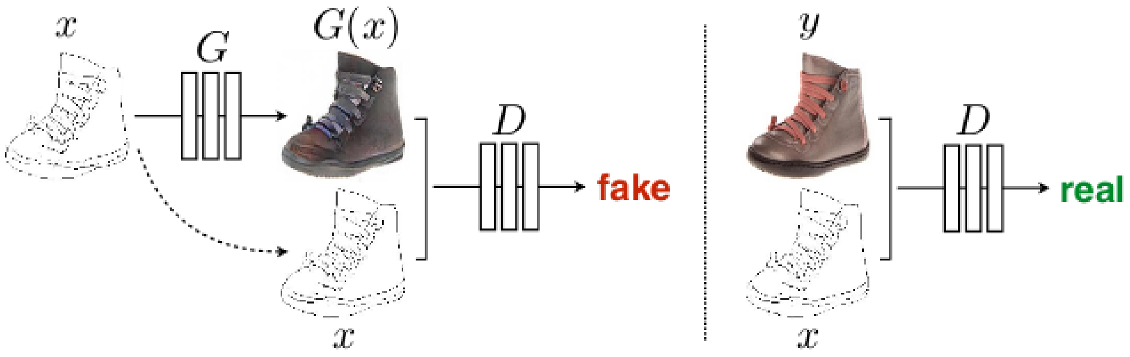

Figure 7: Training a conditional GAN to map edges→photo. The discriminator, D, learns to classify between fake (synthesized by the generator) and real {edge, photo} tuples. The generator, G, learns to fool the discriminator. Unlike an unconditional GAN, both the generator and discriminator observe the input edge map.

Step 4 Read the paper of DRPAN again and again to understand its architecture and master its code for the further work in next week. Practice some examples in Python to enhance the knowledge of this programming language.

There are some differences between GANs and conditional GANs. GANs are generative models that learn a mapping from random noise vector z to output image y, $G : z \rightarrow y$ [2]. In contrast, conditional GANs learn a mapping from observed image x and random noise vector z, to y, $G : \{x, z\} \rightarrow y$. The generator G is trained to produce outputs that cannot be distinguished from "real" images by an adversarially trained discriminator, D, which is trained to do as well as possible at detecting the generators fakes. This training procedure is diagrammed in Figure 7.

The objective of a conditional GAN can be expressed as

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z)))] \tag{1}$$

where G tries to minimize this objective against an adversarial D that tries to maximize it, *i.e.* $G^* = \arg min_G max_D \mathcal{L}_{cGAN}(G, D)$.

To test the importance of conditioning the discriminator, it's also be compared to an unconditional variant in which the discriminator does not observe x:

$$\mathcal{L}_{GAN}(G, D) = \mathbb{E}_{y}[\log D(y)] + \mathbb{E}_{x,z}[\log(1 - D(G(x, z)))] \tag{2}$$

Previous approaches have found it beneficial to mix the GAN objective with a more traditional loss, such as L2 distance. The discriminators job remains unchanged, but the generator is tasked to not only fool the discriminator but also to be near the ground truth output in an L2 sense. It's also explored this option, using L1 distance rather than L2 as L1 encourages less blurring:

$$\mathcal{L}_{L_1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1] \tag{3}$$

The final objective is

$$G^* = \arg \min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L_1}(G) \tag{4}$$

Conditional adversarial networks are a promising approach for many image-to-image translation tasks, especially those involving highly structured graphical outputs. These networks learn a loss adapted to the task and data at hand, which makes them applicable in a wide variety of settings.

5 Plan

Objective:	Run the code of DRPAN then expand it. And modify the experiments to satisfy the article of TPAMI.
Deadline:	2018.11.11

2018.08.06—2018.08.12	Understand and run the code of DRPAN correctly.
2018.08.13—2018.08.19	Based on the origin code, expand the extra experiments.
2018.08.20—2018.08.26	Modify these experiments to satisfy the requirements in article.

References

[1] Pix2pix. `https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix`.

[2] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014.

[3] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[4] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016.

[5] C. Wang, H. Zheng, Z. Yu, Z. Zheng, Z. Gu, and B. Zheng. Discriminative region proposal adversarial networks for high-quality image-to-image translation. *arXiv preprint arXiv:1711.09554*, 2017.